Amendments to the Specification

Please amend page 8, lines 8-9 with the following:

~~Figure 2 illustrates~~ Figures 2A and 2B illustrate a rectangular graphical representation of searching tasks that is used to describe the method if the invention.

Please amend page 8, lines 10-11 with the following:

~~Figure 3 is a multipart figure that provides a detailed example of the application of the method described in Figure 1~~.

Figures 3A through 3Q are multipart figures that provide a detailed example of the application of the method described in Figure 1.

Figure 3A illustrates a graphical representation of an entire task searching 3 sequences against 2 databases.

Figure 3B illustrates a graphical representation of Processor 1 dividing Task 1 vertically (Flowchart Box B); keeps Task 1.A.

Figure 3C illustrates a graphical representation of Processor 1 dividing Task 1.A horizontally (Box C); keeps and begins executing Task 1.A.1.

Figure 3D illustrates a graphical representation of Processor 2 dividing Task 1.B vertically (Box B); keeps Task 1.B.A.

Figure 3E illustrates a graphical representation of Processor 2 dividing Task 1.B.A horizontally (Box C); keeps and begins executing Task 1.B.A.1.

Figure 3F illustrates a graphical representation of Processor 1 completing Task 1.A.1 (Box D); marks it READY (Box A).

Figure 3G illustrates a graphical representation of Processor 2 completing Task 1.B.A.1 (Box D); marks it READY (Box A).

Figure 3H illustrates a graphical representation of Processor 1 completing Task 1.B.A.2 (Box D).

Figure 3I illustrates a graphical representation of Processor 1 merging result for Task 1.B.A.2 with result for Buddy Task 1.B.A.1, thereby computing result for Parent Task 1.B.A (Box E); marks Task 1.B.A READY since Buddy Task 1.B.B. is not READY.

Figure 3J illustrates a graphical representation of Processor 2 completing Task 1.A.2 (Box D).

Figure 3K illustrates a graphical representation of Processor 1 dividing Task 1.B.B horizontally (Box C); keeps Task 1.B.B.1.

Figure 3L illustrates a graphical representation of Processor 2 merging result for Task 1.A.2 with result for Buddy Task 1.A.1, thereby computing result for Parent Task 1.A (Box E); marking Task 1.A. READY, since its Buddy Task 1.B is not ready.

Figure 3M illustrates a graphical representation of Processor 1 completing Task 1.B.B.1 (Box D) marking it READY (Box A) since Task1.B.B.2 is not READY.

Figure 3N illustrates a graphical representation of Processor 2 completing Task 1.B.B.2 (Box D).

Figure 3O illustrates a graphical representation of Processor 2 merging result for Task 1.B.B.1 with result for Buddy Task 1.B.B.2, thereby computing result for Parent Task 1.B.B. (Box E).

Figure 3P illustrates a graphical representation of when Task 1.B.B's Buddy Task 1.B.A is READY, Processor 2 merges result for Task 1.B.B. with result for Buddy Task 1.B.A., thereby computing result for Parent Task 1.B (Box E).

Figure 3Q illustrates a graphical representation of when Task 1.B's Buddy Task 1.A is READY, Processor 2 merges result for Task 1.B with result for Buddy Task 1.A, thereby computing result for Parent Task 1 (Box E). This completes the computation, since Task 1 is the Entire Task and has no Buddy Task.

Please amend page 8, line 12 as follows:

Figure 4 is a timeline that corresponds to the ~~example of Figure 3~~ examples of Figures 3A through 3Q.

Please amend page 8, lines 13-14 as follows:

Figure 5 is a graphical representation of the task division and result merging operations for the example of ~~Figure 3~~ Figures 3A through 3Q.

Please amend page 8, lines 15-18 as follows:

~~Figure 6(a) and 6(b)~~ Figures 6A and B contain graphical comparisons of the performance of the sequence comparison method of the invention running on between 2 and 11 computers, with the performance of the NCBI **BLAST** program running on a single computer of the same type.

Please amend page 12, lines 15-23 as follows:

Figure 1 is a flow chart providing an overview of an example of an implementation of the method of the present invention. Based on the teachings of the instant specification, other implementations would be apparent to the ordinary artisan. Initially, the entire job is a single large Task. Multiple smaller Tasks are created by splitting large Tasks. The new Tasks created by splitting a single Parent Task are called Buddy Tasks. The implementation depicted in Figure 1 interleaves the processes of creating smaller searching tasks, executing those tasks, and merging the results of the smaller searching tasks to create the unified result of the entire

searching task. A VSM bulletin board independent of the worker computers is used to store information about the ongoing searching process. For example, a list of searching tasks (the "Task List") and a list of results of executing the searching tasks (the "Result List") may be stored on the VSM bulletin board.

Please amend page 13, line 28 through page 14, line 8 with the following:

Execution of a searching task requires some quantity of computational resources (*e.g.*, memory, disk, CPU time, etc.), and upon taking a task, a worker computer estimates the quantity of computational resources required to execute the task. This estimate is termed "RES(Task)." RES(Task) is too large if it exceeds the resources available on the computer. If RES(Task) is too large for that particular worker computer, the worker computer will divide the searching task into two smaller searching tasks and add one of them to the Task List kept in the VSM bulletin board. RES(Task) will then be recalculated for the one of the two smaller searching tasks retained by the worker computer. The two smaller searching tasks that are the parts of the now-divided searching task are termed "Buddies." Each new smaller searching task is marked as the other one's Buddy, and the original undivided task is marked as the "Parent" of each of the two new smaller searching tasks.

Please amend page 14, lines 18-22 as follows:

Once the worker computer has a searching task for which neither RES(Task) or GRAN(Task) is too large, it executes the searching task and computes the result for that searching task. The searching task may be executed using any desired algorithm, such as the **BLAST** algorithm. The searching task is termed the worker computer's "Present Task," and the computed result is termed the worker computer's "Present Result." Executing Task creates the corresponding Present Result.

Please amend page 16, lines 10-15 as follows:

~~Figures 2, 3, 4 and 5~~ Figures 2A, 2B, 3A through 3Q, 4, and 5 demonstrate in more detail how the entire searching task is divided up into smaller searching tasks to be performed by each of the computers operating in parallel. Figure 6 contains several charts that illustrate the performance of the method of the present invention in comparison with a standard execution of NCBI **BLAST** on a single computer of the same speed as the worker computers used for the method. As is evident from the charts, a substantial, superlinear speedup may be achieved using the method.

Please amend page 16, lines 16-20 as follows:

As illustrated in ~~Figure 2~~ Figures 2A and 2B, the entire searching task to be performed may be represented by a rectangle, with the horizontal representing the one or more databases against which the query sequences are to be compared, and the vertical representing the query sequences themselves. Any sequence database may be used, such as the sequence databases derived from the databases maintained by the National Center for Biotechnology Information (NCBI). Figure 2A illustrates a representation of one Task searching a single sequence against 2 databases. Figure 2B illustrates a representation of one Task searching 3 sequences against 2 databases.

Please amend page 17, lines 2-9 as follows:

If RES(Task) is too large, a vertical boundary is defined between individual databases or within a database, such as a boundary that most evenly divides the large rectangle representing the undivided searching task into two smaller rectangles. This introduction of a new vertical boundary is illustrated, for example, in ~~Figure 3(b)~~ Figure 3B, where the boundary is introduced between two databases. This process may allow for rearrangement of databases along the

horizontal in order to create a more even division without defining a boundary within a database, or in order to enable a worker computer to create searching tasks that use databases already stored in the memory of the worker computer in question.

Please amend page 17, lines 10-15 as follows:

The method also allows for the individual databases themselves to be split up to permit creation of searching tasks for which RES(Task) is not too large for a given worker computer. This is illustrated in ~~Figure 3(d)~~ Figure 3D. Preferably, the databases are split at defined positions, such as in half, in quarters, etc., so that the results computed for each individual searching task may be more easily merged to provide the unified result. This is not, however, a requirement of the method.

Please amend page 17, lines 16-21 as follows:

The vertical of the rectangle in ~~Figure 2~~ Figures 2A and 2B can be correlated to the relative duration of the task in question, where the duration of any searching task may be measured, for example, by the time in seconds required to execute the searching task with the particular query sequences and the database or portion thereof. The relative duration of the task in question is then equal to the fraction of the duration of the undivided entire searching task represented by the duration of the task in question.

Please amend page 17, line 22-page 18 line 8 as follows:

The first division of the searching task using the estimate of RES(Task), *i.e.*, along the horizontal, splitting up the databases, is related to the quantity of computational resources, such as memory, available on the worker computer. The second division of the searching task, *i.e.*, along the vertical, by splitting up the query sequences, as illustrated in ~~Figure 3(c)~~ Figure 3C, is related to the estimated relative duration of the searching task. In order to obtain the largest

possible speedup, the method tries to ensure (1) that executed searching tasks are small enough, *i.e.*, of sufficiently *short* relative duration, so that there will be enough tasks to fully occupy all of the worker computers available for the entire searching task, and (2) that executed searching tasks are large enough, *i.e.*, of sufficiently *long* duration, that the amount of overhead related to the use of parallelism (*i.e.*, the costs related to communication, access to the VSM, and task startup or shutdown on the individual worker computers) are small enough that the overall method is efficient. By ensuring these two properties, the method is able to achieve linear speedup attributable to the full and efficient use of all of the worker computers available to perform the entire searching task. In practice, however, the method often achieves superlinear speedup because the divisions based on RES(Task) reduce the amount of I/O overhead, which leads to additional speedup beyond the linear speedup that would be expected normally.

Please amend page 21, lines 15-21 as follows:

As noted above, Figure 1 is a flowchart depicting an example of implementing the instant method, which interleaves the processes of task division, task execution, and creation of the unified result for the entire searching task. This implementation of the creation of the unified result for the entire searching task may be more efficient than the simple implementation just described because it involves less overhead related to the use of the VSM bulletin board. The detailed example of ~~Figure 3 illustrates~~ Figures 3A through 3Q illustrate the sequence of task divisions and result mergings that might be achieved by the method using the implementation of Figure 1.

Please amend page 26, lines 3-line 10 with the following:

~~Figure 3 provides~~ Figures 3A through 3Q provide a detailed example of the application of the method of the instant invention using the implementation of Figure 1. Each of Figures 3A

through 3Q shows the representation of the entire searching task at a particular time point during a sample operation of the method of the invention when run on two processors. In addition to the representation of the tasks, Figures 3A through 3Q also show the contents of 2 important lists on the bulletin board (i.e., the Task List and the Result List) and indicates the current activities for each of the two participating processors at the corresponding instant of time. The Entire Task is "Task 1". Tasks created by splitting larger divisions are denoted by names using dotted notation in which either the Parent Task's name is extended with a period (".") followed either by a capital letter or an Arabic numeral. Capital letters are used when vertical splitting is performed based on RES(Task), as when Task 1.A and Task 1.B denote the two tasks created by subdividing Task 1. Arabic numerals are used when horizontal splitting is performed based on GRAN(Task), as when Task 1.A.1 and Task 1.A.2 denote the two tasks created by subdividing Task 1.A. The computation in question entails the searching of a group of query sequences against two databases using two processors (i.e., worker computers). Each of ~~the lettered subfigures of Figure 3 (i.e., Figure 3(a) through Figure 3(q) inclusive)~~ Figure 3A through Figure 3Q is a representation of the state of the computation at a particular instant in time. (Figure 4 contains timelines showing the activities of the processors between the time points that correspond to the subfigures. Figure 4 also contains lettered markings that correlate to ~~subfigures of Figure 3~~ Figures 3A through 3Q to their specific points in time during the computation.) The processor activities are correlated with Figure 4, which illustrates the details of the processor activity and includes a time line that is correlated to Figures 3A through 3Q.

Please amend page 26, lines 11-15 as follows:

~~Each subfigure of Figure 3~~ Each of Figures 3A through 3Q contains four sections reflective of the states of the searching task, the processors and the VSM bulletin board at the time point in question:

(1) A rectangular representation similar to those of ~~Figure 2~~ Figures 2A and 2B that represents the entire searching task as subdivided into smaller searching tasks at the time point in question;

Please amend page 26, line 22 thorough page 27, line 10 as follows:

The Legend included in ~~Figure 3~~ Figures 3A through 3Q describes the graphical markings and the Task naming conventions used in the example. Similar markings are used in Figures 4 and 5, as well.

~~Figure 3 illustrates~~ Figures 3A through 3Q illustrate the most important operations in the method using the implementation of Figure 1:

(1) ~~Figures 3(b) and 3(d)~~ Figures 3B and 3D illustrate the division of tasks by dividing and/or rearranging one or more databases (i.e., represented as the introduction of a new vertical boundary).

(2) ~~Figures 3(c), 3(e) and 3(k)~~ Figures 3C, 3E and 3K illustrate the division of tasks by dividing the query sequences (i.e., represented as the introduction of a new horizontal boundary).

(3) ~~Figures 3(f), 3(g) and 3(m)~~ Figures 3F, 3G and 3M illustrate the result of executing a task which has a Buddy Task that is not READY.

(4) ~~Figures 3(i) and 3(l)~~ Figures 3I and 3L illustrate the case of performing a single merging step that leads to a unified task for which no further unification

is possible until other tasks have been completed. (Such tasks are marked as READY and placed on the Result List in the VSM bulletin board.)

(5) ~~Figure 3(o), 3(p) and 3(q)~~ Figures 3O, 3P, and 3Q illustrate the case of repeated hierarchical merges that eventually lead to the final result for the entire searching task.

Please amend by inserting at page 27, line 11 the following:

Figure 4 contains timelines that illustrate the activities carried out on each of two processors during application of the method of the invention to compute the result of the entire searching task as illustrated in Figures 3A through 3Q. The markings for each activity are described below. In this figure, the fill pattern for each activity reflects the type of activity. The time scale does not represent actual time, but is intended to portray possible relative times at which various activities might take place. The time scale is consistent with the details of Figures 3A through 3Q and with a possible operation of actual computer software implementing the method. The timelines are correlated with Figures 3A through 3Q.

Please amend page 27, lines 11-19 as follows:

To complete the picture of the example of ~~Figure 3~~ Figures 3A through 3Q, Figure 5 illustrates the task division and result merging operations using a binary tree representation. In Figure 5, each division of a searching task into two smaller searching tasks is represented by a single white rectangle (representing the searching task to be divided) containing two outward-pointing arrows, each of which leads to a smaller white rectangle representing one of the two smaller searching tasks. The parenthesized letters refer to Figures 3A through 3Q. Task names also refer to the names used in Figures 3A through 3Q. Analogously, the creation of a unified result for a Parent Task by merging the computed results of two Buddy Tasks is represented by

two gray rectangles (the Buddy tasks) connected by outward-pointing arrows to a single larger gray rectangle (the Parent task). As with the other figures, Figure 5 contains lettered markings to correlate it with ~~the subfigures of Figure 3~~ Figures 3A through 3Q.

Please amend Figures 1 through 6 as illustrated in the attached replacement drawings.